論文2001-38SP-5-6

비트 플레인 정합에 의한 움직임 추정기의 VLSI 설계

(VLSI Design for Motion Estimation Based on Bit-plane Matching)

高 永 基 *, 吳 亨 哲 **, 高 聖 濟 ***

(Young-Ki Ko, Hyeong-Cheol Oh, and Sung-Jea Ko)

요 약

전역탐색알고리즘(full-search algorithm, FSA)은 탐색영역의 범위가 커짐에 따라 방대한 양의 계산을 필요로 하기 때문에 이에 따른 알고리듬의 처리시간이 커지고, 하드웨어로 구현했을 때 회로가 복잡해진다는 문제점을 안고 있다. 본 논문에서는 이러한 문제점을 개선하기 위한 방안으로 비트플레인 정합에 의한 움직임 추정기의 VLSI 구조를 제안한다. 제안된 움직임 추정기에서는 비트 플레인 정합기준을 이용하여 기존의 전역 탐색 알고리즘을 하나의 이진영상으로 적용함으로써 움직임 추정에 소요되는 연산의 양을 크게 줄이면서도 전역탐색 알고리듬과 유사한 움직임 추정 성능을 갖도록 하였으며, 제안된 VLSI 구조에서는 두 개의 프로세싱 코어를 채택하여 데이터 흐름을 시스톨릭 (systolic) 어레이의 형태로 제어하여, 시스템 내부의 SRAM을 제거하여 동작 속도 상의 이득뿐만 아니라, 메모리 공정을 필요로 하지 않는 저가의 공정을 사용 가능하게 함으로써 제작상의 비용을 절감할 수 있는 해결책을 제시하였다. 구현된 하드웨어는 VHDL을 이용하여 설계하고, 기능 검증을 수행한 후 0.6-μm three-metal CMOS 공정을 이용하여 $8.15\times10.84mm^2$의 크기로 집적하였다.

Abstract

Full-search algorithm requires large amount of computation which causes time delay or very complex hardware architecture for real time implementation. In this paper, we propose a fast motion estimator based on bit-plane matching, which reduce the computational complexity and the hardware cost. In the proposed motion estimator, the conventional motion estimation algorithms are applied to the binary images directly extracted from the video sequence. Furthermore, in the proposed VLSI motion estimator, we employ a pair of processing cores that calculate the motion vector continuously. By controlling the data flow in a systolic fashion using the internal shift registers in the processing cores, we avoid using SRAM (local memory) so that we remove the time overhead for accessing the local memory and adopt lower-cost fabrication technology. We modeled and tested the proposed motion estimator in VHDL, and then synthesized the whole system which has been integrated in a 0.6-μm triple-metal CMOS chip of size $8.15\times10.84mm^2$.

* 正會員, LG 전자
(LG Electronics Co., Ltd.)
** 正會員, 高麗大學校 電子 및 情報工學科
(Dept. of Electronics and Information Engineering, Korea University)
*** 正會員, 高麗大學校 電子工學科
(Dept. of Electronics Engineering, Korea University)
接受日字:2001年2月1日, 수정완료일:2001年7月22日

Ⅰ. 서 론

최근 들어 디지털 비디오 압축에 대한 깊은 관심과 많은 연구가 진행되고 있다. VOD(video on demand), DTV(digital TV)/HDTV (high definition TV) 방송, 디지털 동영상 저장 매체, 화상회의, 화상전화와 같은 시스템들은 이러한 디지털 비디오 압축 기술을 사용하는 시스템들로 널리 알려져 있다. 디지털 비디오 압축 기술들 중에서 움직임 추정 기술은 상호 인접 프레임 간의 움직임을 추정하여 시간적인 정보의 중복성을 제거하고 고압축을 실현하는 핵심 기술이며, 동영상 압축 부호화의 표준인 ITU의 H.261/263이나 ISO의 MPEG (motion picture expert group)에서 국제 표준 기술로 채택하여 사용하고 있다[1~3].

움직임 추정 기법은 크게 나누어 화소 단위의 움직임 추정 기법인 PRA(pel-recursive algorithm)와 일정한 크기의 블록 단위의 움직임 추정 기법인 블럭정합 알고리즘(block matching algorithm, BMA)로 구분할 수 있다[4]. PRA는 매 해당 화소마다 방대한 연산량을 필요로 하기 때문에 이를 실제 시스템에 응용하기는 현실적으로 불가능하며, 화소들을 일정 크기로 묶어서 이들이 동일한 움직임을 갖는다는 가정 하에 움직임을 추정하는 BMA가 주로 사용된다. 이때 블록의 움직임은 MSE (mean square error)나 MAD(mean absolute difference)와 같은 상관 계수가 최대/최소가 되는 지점을 찾음으로써 추정할 수 있다.

BMA 중에서 FSA는 블록의 움직임을 추정하는데 있어서 탐색블럭 내의 모든 가능한 후보 점들에서 기준블럭과 해당블럭을 비교하여 움직임 벡터(motion vector, MV)를 추정하기 때문에 움직임 추정 범위가 증가함에 따라 방대한 연산량을 필요로 하게되어 처리 시간이 길어지고, 하드웨어로 구현하였을 때 회로가 복잡해지는 문제점을 안고 있다[4,12~14]. 이러한 문제점을 해결하기 위한 방안으로 영상으로부터 경사값(gradient)이나 에지 등을 추출하고 이로부터 움직임을 추정하는 기법들이 제안되었으나 이러한 정합 방법은 이진 영상을 생성하기 위한 별도의 하드웨어를 요구하는 문제점과 함께 추정 결과가 FSA에 비해 부정확한 단점을 갖고 있다[8,9]. 또한 퓨리에 변환이나 DCT (discrete cosine transform) 등을 이용하여 주파수 영역에서 움직임을 추정하는 기법들이 제안되었지만, 많은 계산량을 필요로 한다는 점에서는 비슷한 문제점을 안고 있다[10]. 기존의 발표된 논문 [5]에서는 블록정합 기준을 계산하기 위해 256 레벨의 화소값을 표현할 수 있는 8비트 중 상위 비트들을 하나의 군으로 묶어서 최소의 비트들만을 이용한 움직임추정 방법이 제안되었다. 하지만 [5]에서는 FSA와 유사한 성능을 갖기 위해서는 8비트 중에서 상위 4비트가 최적의 성능을 보임을 알 수 있다.

본 논문에서는 비트 플레인(bit-plane) 정합을 이용한 움직임 추정기의 VLSI 구조를 제안한다[6]. 제안된 움직임 추정기에서는 비트 플레인으로부터 추출한 이진 영상을 기존의 전역 탐색 알고리즘에 적용함으로써 움직임 추정에 소요되는 계산량을 크게 감소시켰다. 제안된 움직임 추정기의 VLSI 구조가 갖는 가장 큰 장점은 움직임 추정에 이진 영상을 이용하기 때문에 기존의 알고리즘의 산술 연산들을 단순한 이진 논리 연산으로 처리함으로써 처리 시간을 크게 줄일 수 있다는데 있다. 또한 제안된 VLSI 구조에서는 두 개의 프로세싱 코어를 채택하여 데이터 흐름을 시스톨릭 (systolic) 어레이의 형태로 제어하여, 시스템 내부의 탐색영역의 데이터를 저장하기 위한 국부 메모리인 SRAM을 제거하여 동작 속도 상의 이득뿐만 아니라, 메모리 공정을 필요로 하지 않는 저가의 공정을 사용 가능하게 함으로써 제작상의 비용을 절감할 수 있는 해결책을 제시하고 있다.

본 논문의 구성은 다음과 같다. Ⅱ장에서는 비트 플레인 정합 기준을 이용한 움직임 추정 방식에 대하여 설명하고, Ⅲ장에서는 제안된 VLSI의 구조에 대하여 설명한다. Ⅳ장에서는 구현 시스템에 대한 실험 결과를 제시하고, 실험 결과에 대한 성능을 분석한다. Ⅴ장에서 결론을 맺는다.

Ⅱ. 비트 플레인을 이용한 움직임 추정 기법

비트 플레인 영상으로부터 생성한 이진 영상을 이용한 움직임 추정 기법은 기존의 그레이 영상에서 움직임 추정에 필요한 산술 연산들을 간단한 이진 논리 연산으로 대처할 수 있다는데 장점을 갖는다. 그레이 영상으로부터 에지를 추출한 후 이를 이진 영상으로 변환한 후, 이로부터 영상의 전역적인 움직임을 추출하여 움직임 추정 시스템을 구현한 예가 발표된 바 있으나[7,8], 이는 에지 영상을 생성하기 위한 별도의 계산이 필

요하며 에지 추출의 정확도에 따라 움직임 추정의 성능이 달라지는 단점을 모두 가지고 있다.

제안된 움직임 추정기에서는 움직임 정합에 사용할 데이터를 비트플레인으로부터 바로 생성하기 때문에 부가적인 하드웨어와 계산을 필요로 하지 않으며, 적절한 비트 플레인을 선정한다면 에지 검출 방식 등에 비해 원영상의 전체적인 윤곽과 세부적인 정보를 동시에 반영할 수 있는 이진 영상을 생성할 수 있다는 장점을 갖는다. 비트 플레인 정합기준에 의한 움직임 추정 기법을 설명하기에 앞서서 비트 플레인 영상을 생성하는 방법에 대해 알아보자.

1. 비트 플레인 영상의 생성 및 특성

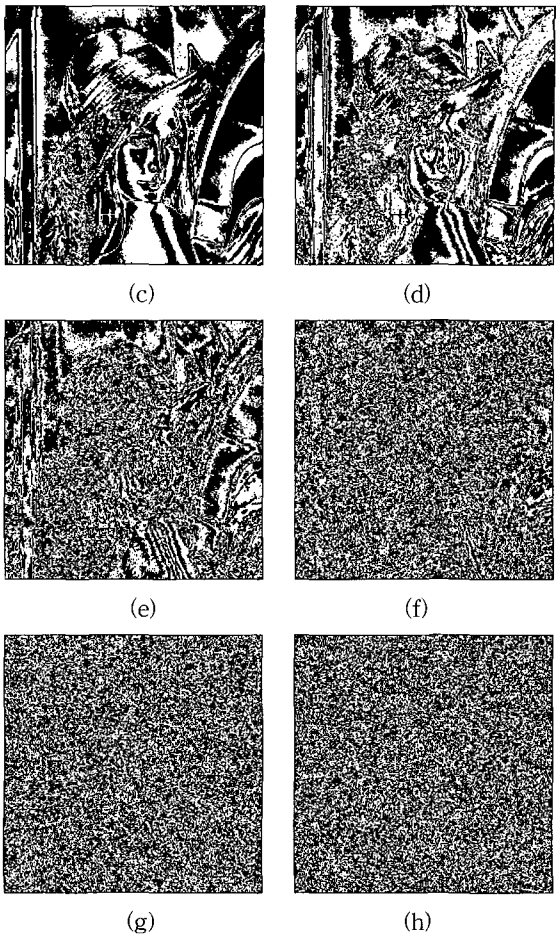
2^{K+1} 의 그레이 레벨 해상도와 $L\times L$ 공간 해상도를 갖는 동영상을 f로, 그리고 시각 t에서의 현재 프레임과 그 이전 프레임은 각각 f^t 와 f^{t-1} 로 나타내기로 하자. 그러면 현재 프레임 내의 (i, j) 번째 화소는 다음과 같이 $K+1$ 비트로 표현된다.

$$f^t(i, j) = a_K 2^K + a_{K-1} 2^{K-1} \cdots + a_0 \qquad (1)$$

이 때 각 화소를 이루는 $K+1$ 비트의 데이터들 중에서 동일한 위치의 비트들만을 모아서 하나의 이진 영상을 생성할 수 있는데, 이를 비트 플레인 영상이라 한다. 따라서 2^{K+1} 의 그레이 레벨 해상도를 갖는 영상으로부터는 $K+1$ 개의 비트 플레인 영상을 생성할 수 있다. 그림 1에 256 그레이 레벨 해상도를 갖는 Lena 영상으로부터 8개의 비트 플레인 영상을 생성한 예를 보였다.

그림 1에 보인 각 비트 플레인들이 원래의 그레이 영상과 어떤 관계를 갖는지 알아보자. 가장 상위의 2^7 비트 플레인의 의미는 쉽게 알 수 있는데, 이는 그레이 영상을 256 그레이 레벨의 중간인 128에서 문턱값을 적용한 결과이다. 즉 f^t 의 2^k 비트 플레인을 $b_k{}^t$, $(0 \le k \le K)$로 나타내기로 하면, 최상위 비트 플레인 $b_K{}^t$ 는 다음과 같다.

(a) (b)



(c) (d) (e) (f) (g) (h)

그림 1. 비트 플레인 영상의 예, (a)~(h) 2^7~2^0 비트 플레인 영상

Fig. 1. Decomposition of eight bit-plane images from a grayscale image: (a)~(h) 2^7~2^0 bit-plane image.

$$b_K{}^t(i, j) = \begin{cases} 1 & \text{if } f^t(i, j) \ge 2^K, \\ 0 & \text{otherwise.} \end{cases} \qquad (2)$$

그림에서 (a)~(d)는 Lena 영상에 대한 상위 비트 플레인들로써 영상의 전체적인 윤곽을 나타내는 것을 알 수 있다. 그림의 (e)~(h)에 나타낸 것과 같이 하위 비트 플레인으로 내려갈수록 영상의 디테일이 포함되기 시작하여 그림 1의 (h)와 같은 최하위 비트 플레인에서는 너무 세밀한 영상 정보만을 포함하고 있기 때문에 육안으로는 거의 의미를 찾아보기 어렵다. 따라서 움직임 추정의 관점에서 각 비트 플레인들을 고려하여 본다면, 하위 비트 플레인들은 영상들 간의 상관성을 찾아보기 힘들기 때문에 움직임 추정에는 적합하지 않으

며, 시각적으로 영상의 윤곽과 세부정보를 동시에 포함하는 상위 비트 플레인을 선정하는 것이 움직임 추정에 적합하리라는 것을 알 수 있다.

2. 비트 플레인 정합을 이용한 움직임 추정

본 절에서는 그레이 영상으로부터 생성한 이진 영상, 즉 비트 플레인을 이용한 움직임 추정 기법을 제시한다. 제안된 기법은 기존의 대표적인 움직임 추정 기법인 BMA에 기반하고 있다. 본 논문에서 제안하는 움직임 추정 알고리즘의 처리 과정은 상관 연산기의 입력으로 이진 영상을 이용한다는 점을 제외하고는 기존의 BMA의 처리 과정과 거의 유사하다. 그림 2는 비트 플레인 정합 기준에 의한 제안된 움직임 추정 알고리듬의 처리 과정을 보여 주고 있다.

비트 플레인 정합을 이용한 움직임 추정 시스템에서 비트 플레인 영상을 생성하는 블록은 프레임 메모리로부터 입력되는 현재 프레임과 이전 프레임에서 특정 비트 플레인을 선택하여 이진 영상을 생성한다. 이 과정에서는 각 화소들의 그레이레벨 데이터로부터 한 비트만을 추출하기 때문에 별도의 하드웨어를 필요로 하지 않는다. 이렇게 생성된 이진 영상들로부터 상관 계수을 계산하고, 최대 상관도를 보여주는 위치를 찾아 이를 움직임 벡터(motion vector, MV)로 출력한다. 움직임 벡터를 결정하기 위한 상관 계수로는 식 (3)과 같은 k 번째 비트 레벨에서의 XOR 연산의 합을 사용한다.

$$C(m,n)=\sum_{i=1}^{N}\sum_{j=1}^{N} b_k^{t}(r_x+i, r_y+j) \oplus b_k^{t-1}(r_x+m+i, r_y+n+j) \quad (3)$$

식 (3)에서 $b_k^{t}(\cdot)$ 는 앞에서 설명한 바와 같이 동영상 내의 임의의 화소 $f^{t}(\cdot)$ 의 k 번째 비트 플레인 영상을, 그리고 (r_x, r_y) 와 (m, n) 는 각각 프레임 상에서의 기준 블록의 위치와 탐색 영역 내에서의 화소의 위치를 가리킨다.

움직임 추정의 입장에서 제안된 추정기의 움직임 추정 능력을 고려할 때 하나의 단일 비트플레인을 움직임 추정에 이용하기 때문에, 움직임 추정기의 성능 향상을 위해서는 적절한 비트 플레인을 선정하는 것이 매우 중요하다. 본 논문에서 제안하는 움직임 추정기에서는 최적의 움직임 추정성능을 나타내는 비트 플레인을 선정하기 위해 "Flower Garden", "Mobile Calendar" 그리고 "Table Tennis"와 같은 표준 테스트 시퀀스들을 이용하여 각각의 비트 레벨에 대한 움직임 성능을 PSNR(peak-peak to signal to noise ratio) 측면에서 비교하여 보았다. 성능을 평가하기 위해 기준블럭의 크기를 16×16 으로 설정하고 탐색영역의 크기를 수평·수직 각각의 방향으로 [-16, 15]으로 추정 범위를 설정하고, 각 비트레벨에 대한 움직임 추정 성능을 PSNR 측면에서 비교하였다. 성능 비교 결과 6번째 비트 플레인을 이용한 움직임 추정 성능이 모든 영상 시퀀스에 대해 안정적인 성능을 나타내었다. 따라서 제안된 움직임 추정기에서는 6번째 비트 플레인을 이용하여 움직임 추정을 수행하도록 하였다.

Ⅲ. 제안된 VLSI 구조

본 논문에서 제안된 움직임 추정 알고리즘을 이용한 움직임 추정기는 기준 블록에 대한 크기를 16×16 화소로 설정하고, 움직임 추정 범위를 수평·수직 방향으로 [-16, 15]로 지정하였다. 32개의 단일처리기(processing element, PE)로 구성된 두 개의 프로세싱 코어를 설정하여 현재 프레임의 각각 다른 두 개의 기준블럭, 즉 우수 기준블럭(even reference block)과 기수 기준블럭(odd reference block)에 대한 MV를 계산하도록 하였고, 각각에 대한 상관 계수인 XOR의 합을 연산하기 위해서 PE를 최대로 활용할 수 있게 하여 병렬처리의 효과를 최대로 활용하였다. 그림 3은 제안된 움직임 추정기에 대한 VLSI 구조를 보인 것이다. 그림에서 align buffer는 프레임메모리부터 입력된 탐색영역의 데이터를 47 비트의 단위로 프로세싱코어로 전달하기 위한 버퍼이며, RBR (reference block register)은 기준 블록을 저장하기 위한 레지스터이다. 프로세싱 코어 내에 설정된 쉬프트레지스터(shift registers, SR's)들은 탐색영역의 1/3에 해당하는 데이터를 저장할 수 있도록 설정하여, 프로세싱 코어로 데이터를 전달하여 XOR의 합을 계산하도록 한다. 두 개의 프로세싱 코어에 탐색 영역의 데이터가 저장된 후에는 두 개의 프로세싱 코어가 탐색영역의 데이터가 연속적으로 이용하여 기수 기준블록과 우수 기준블록에 대한 움직임 벡터를 동시에 출력하도록 설계되어 있다. 두 개의 독립적인 프로세싱 코어가 동시에 동작하게 되어 시스템의 처리속도가 빨라지고 프로세싱 코어의 데이터 처리율이 증가하게 된다. 또한 프로세싱 코어 내부의 SR은 데이터 흐름을 시스톨릭(systolic) 형태로 제어함으

로써 기존의 움직임 추정 시스템에 사용되는 국부 메모리(local memory)인 SRAM을 제거할 수 있도록 하고 있다. 따라서 본 논문에서 제안된 움직임 추정기의 VLSI 구조는 국부 메모리를 억세스하기 위해 부가적으로 소요되는 시간적인 오버헤드를 줄일 수 있을 뿐만 아니라, 메모리 공정을 사용하지 않는 저가의 공정을 가능하게 하여 공정 비용을 줄일 수 있는 구조를 갖고 있다.

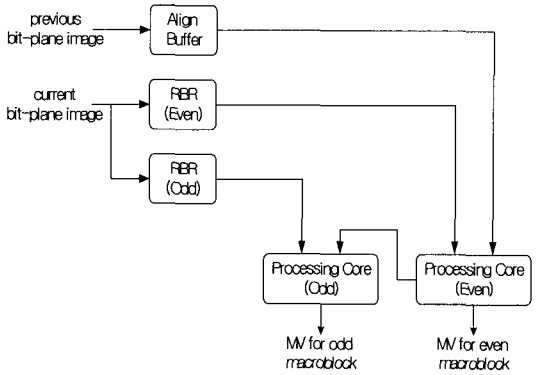


그림 3. 제안된 시스템에 대한 VLSI 구조
Fig. 3. Block diagram of the proposed architecture.

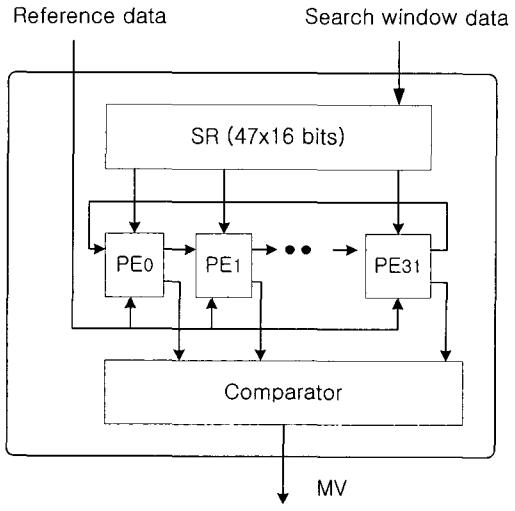


그림 4. 프로세싱 코어에 대한 블록도
Fig. 4. Block diagram of the proposed processing core.

그림 4는 제안된 움직임 추정기의 프로세싱 코어에 대한 블록도를 나타낸 것이다. 그림에서 프로세싱 코어는 각각의 탐색 영역의 탐색점에서 비트 플레인 정합 기준에 사용되는 상관계수인 XOR 연산결과를 출력하기 위한 부분으로 시스톨릭 어레이를 기반으로 하는 일차원 어레이의 구조로 되어 있다. 프로세싱 코어 내의 일차원 어레이의 구조로 구성된 단일처리기(processing element, PE)들은 16개의 화소를 동시에 비교하고 이에 대한 상관계수의 값을 계산하기 때문에 기존의 일차원 구조의 움직임 추정기에 비해 계산시간을 감소시킬 수 있는 장점을 갖는다.

그림 5는 PE의 구조를 보인 것이다. PE는 16개 화소를 동시에 계산하기 위한 16 XOR 논리회로와 XOR의 결과를 누적하기 위한 덧셈기로 구성되어 있다. 덧셈기의 구현방법에 따라서 시스템의 속도를 향상시킬 수 있는데, 본 논문에서는 단일처리기의 덧셈기에 대한 배치를 연산 후 발생하는 캐리와 전단의 단일처리기의 값을 더하는 형태로 설정하여 덧셈기의 캐리와 전단의 상관계수의 값을 적절히 처리하도록 하여 시스템의 속도에 영향을 줄 수 있는 경로를 줄일 수 있는 방안으로 설계하였다.

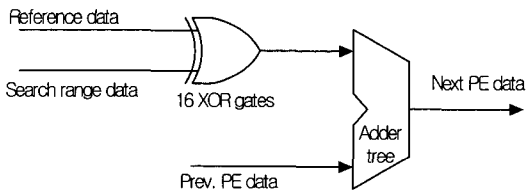


그림 5. 단일처리기에 대한 블록도
Fig. 5. Block diagram of the proposed processing element.

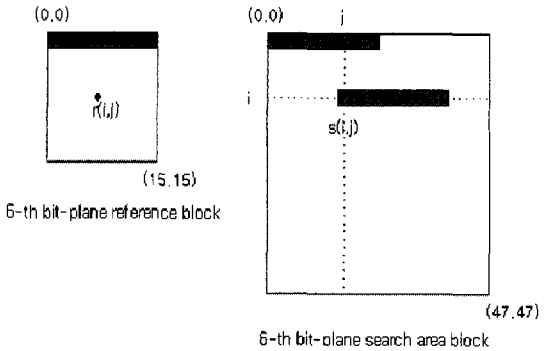


그림 6. 탐색 영역과 기준 블록에 대한 픽셀 좌표
Fig. 6. Pixel coordinate for reference block and search area data.

그림 6과 표 1는 현재 프레임의 6번째 비트 플레인 영상에서 기존 영역의 데이터와 이전 프레임의 탐색 영역의 데이터가 프로세싱 코어로 전달되는 형태을 나

표 1. 프로세싱 코어의 데이터 흐름 제어
Table. 1. Basic data flow for the proposed processing core.

time	PE_0	PE_1	···	PE_{14}	PE_{15}	···	PE_{31}
0	$R_0, S_{i,0}$	$R_0, S_{i,1}$	·	$R_0, S_{i,14}$	$R_0, S_{i,15}$	·	$R_0, S_{i,31}$
1	$R_1, S_{i,32}$	$R_1, S_{i,1}$	·	$R_1, S_{i,14}$	$R_1, S_{i,15}$	·	$R_1, S_{i,31}$
2	$R_2, S_{i,32}$	$R_2, S_{i,33}$	·	$R_2, S_{i,14}$	$R_2, S_{i,15}$	·	$R_2, S_{i,15}$
·	·	·	·	·	·	·	·
·	·	·	·	·	·	·	·
14	$R_{14}, S_{i,32}$	$R_{14}, S_{i,33}$	·	$R_{14}, S_{i,14}$	$R_{14}, S_{i,15}$	·	$R_{14}, S_{i,31}$
15	$R_{15}, S_{i,32}$	$R_{15}, S_{i,33}$	·	$R_{15}, S_{i,14}$	$R_{15}, S_{i,15}$	·	$R_{15}, S_{i,31}$
16+0	$R_0, S_{i+1,0}$	$R_0, S_{i+1,1}$	·	$R_0, S_{i+1,14}$	$R_0, S_{i+1,15}$	·	$R_0, S_{i+1,31}$
16+1	$R_1, S_{i+1,32}$	$R_1, S_{i+1,1}$	·	$R_1, S_{i+1,14}$	$R_1, S_{i+1,15}$	·	$R_1, S_{i+1,31}$
16+2	$R_2, S_{i+1,32}$	$R_2, S_{i+1,33}$	·	$R_2, S_{i+1,14}$	$R_2, S_{i+1,15}$	·	$R_2, S_{i+1,15}$
·	·	·	·	·	·	·	·
·	·	·	·	·	·	·	·
16+14	$R_{14}, S_{i+1,32}$	$R_{14}, S_{i+1,33}$	·	$R_{14}, S_{i+1,14}$	$R_{14}, S_{i+1,15}$	·	$R_{14}, S_{i+1,31}$
16+15	$R_{15}, S_{i+1,32}$	$R_{15}, S_{i+1,33}$	·	$R_{15}, S_{i+1,14}$	$R_{15}, S_{i+1,15}$	·	$R_{15}, S_{i+1,31}$
·	·	·	·	·	·	·	·
·	·	·	·	·	·	·	·
31×16+0	$R_0, S_{i+31,0}$	$R_0, S_{i+31,1}$	·	$R_0, S_{i+31,14}$	$R_0, S_{i+31,15}$	·	$R_0, S_{i+31,31}$
31×16+1	$R_1, S_{i+31,32}$	$R_1, S_{i+31,1}$	·	$R_1, S_{i+31,14}$	$R_1, S_{i+31,15}$	·	$R_1, S_{i+31,31}$
31×16+2	$R_2, S_{i+31,32}$	$R_2, S_{i+31,33}$	·	$R_2, S_{i+31,14}$	$R_2, S_{i+31,15}$	·	$R_2, S_{i+31,15}$
·	·	·	·	·	·	·	·
·	·	·	·	·	·	·	·
31×16+14	$R_{14}, S_{i+31,32}$	$R_{14}, S_{i+31,33}$	·	$R_{14}, S_{i+31,14}$	$R_{14}, S_{i+31,15}$	·	$R_{14}, S_{i+31,31}$
31×16+15	$R_{15}, S_{i+31,32}$	$R_{15}, S_{i+31,33}$	·	$R_{15}, S_{i+31,14}$	$R_{15}, S_{i+31,15}$	·	$R_{15}, S_{i+31,31}$

타낸 것이다. 그림에서 표기된 $r(i,j)$는 16×16 기준 블럭의 화소 위치를 $s(i,j)$는 47×47 탐색 영역에서의 화소 위치를 나타낸다. 표는 32개의 탐색 점에서 MV가 계산되는 과정에서 기준블럭의 데이터와 탐색 영역의 데이터가 어떻게 처리되는지를 정리한 것이다. 표에서 기준블럭과 탐색영역의 i번째 열에 해당하는 16개의 화소를 각각 $R_i=[r_{i,0}, r_{i,1}, \cdots r_{i,15}]$, $S_{i,j}=[s_{i,j}, s_{i,j+1}, \cdots s_{i,j+15}]$로 표현하였다. 표에 나타낸 것과 같이 시간 $t=0$에서는 기준 블럭의 첫 번째 열인 R_0와 이에 해당하는 탐색영역의 데이터인 $S_{i,j}$가 프로세싱 코어로 입력된다. 이러한 데이터 제어 과정에 의해 첫 32개의 탐색점에 대한 상관계수가 $t=15$에서 계산되며, $t=511$에서는 탐색 영역 내의 모든 탐색점에 대한 상관계수를 계산하여 상관계수가 최소인 위치를 찾아내고, 그 위치를 MV로 출력한다.

Ⅳ. 성능 평가 및 분석

제안된 움직임 추정 기법의 성능을 평가하기 위해서 전역탐색(full-search algorithm, FSA)[4], 삼단탐색(three-step search, TSS)[7], 에지 검출된 이진영상을 이용한 움직임 추정(motion estimation via one-bit transform, MEOBT)[8]과 같은 여러 기법들과 제안된 비트 플레인 정합에 의한 움직임 추정기법의 성능을 모의 실험을 통해 비교하여 보았다. 모의실험에서는 2장에서 사용한 표준 영상 시퀀스인 "Flower Garden", "Mobile Calendar" 그리고 "Table Tennis" 이용하여 평균 PSNR의 값을 비교하였다. 표 2에 움직임 추정 성능에 대한 모의 실험결과를 요약하였다. 표에 보인 것과 같이, FSA와 제안된 기법의 성능 비교에서는 움직임이 상대적으로 적은 테스트 시퀀스에서는 유사한

성능을 나타내는 것을 알 수 있고, TSS, MEOBT와 같은 고속 탐색 알고리듬과의 비교에서는 거의 유사한 성능을 갖는 것을 알 수 있다.

표 2. 테스트 시퀀스를 이용한 각 알고리듬의 평균 PSNR 비교(dB)
Table. 2. Average PSNR performance between different methods in each sequence (dB)

알고리듬	"Mobile Calendar"	"Flower Garden"	"Table Tennis"
FSA	23.69	24.21	28.51
TSS	23.51	23.60	27.34
MEOBT	23.29	23.44	27.21
Proposed	23.39	23.59	27.59

기존에 발표된 여러 움직임 추정기의 VLSI 구조에서는 탐색영역의 데이터를 정형화된 systolic array 구조를 채택한 논문들이 발표되었다[11~14]. 움직임 추정기에 대한 VLSI 구조에서 탐색영역 데이터의 흐름을 조절하여 동작 속도를 향상시키거나, 메모리의 대역폭이나 I/O 핀 수를 줄이는 목적의 논문이 발표되었다[14]. 그러나 기존의 제안된 논문들은 참조영역을 저장하기 위한 부가적인 메모리를 필요로 하기 때문에 참조영역의 데이터를 불러오거나 저장하기 위한 메모리 억세스 시간과 그레이 영상 즉, 8비트 영상을 이용하여 움직임을 추정하기 때문에 이에 따른 계산상의 과도한 연산량을 요구하는 문제점을 안고 있다. 본 논문에서는 이러한 문제점을 해결하기 위해서 두 개의 프로세싱 코어를 이용하여 참조영역을 저장하기 위한 메모리를 제거하고, 비트 플레인 정합을 이용하여 움직임 추정에 사용되는 데이터를 하나의 단일 비트로 표현하여 처리하였다. 표 3은 제안된 VLSI 구조와 기존의 발표된 VLSI 구조를 비교 정리한 것이다. 표에서 알 수 있듯이, 제안된 구조에서는 두 개의 프로세싱 코어를 이용하여 참조영역을 저장하기 위한 메모리를 제거하여 메모리에 저장된 데이터를 처리하는 시간을 절약하고, 프로세싱 코어의 사용 효율을 높여서 단위시간당 두 개의 MV를 출력하는 효율적인 구조로 되어 있음을 알 수 있다.

제안된 움직임 추정기는 VHDL을 이용하여 모델링하였고, 모델링된 하드웨어는 FPGA 상에서 기능 검증을 거친 후, 0.6 μm triple-metal 공정을 이용하여 제작하였다. 그림 7은 칩의 레이 아웃을 보여주는 것으로 전체 시스템의 칩에 대한 게이트의 수는 2-input NAND gate를 기준으로 약 10만 4천 개이며, 칩의 크기는 $8.15\times10.84\,mm^2$이다. 제안된 VLSI 구조에서는 두 개의 프로세싱을 코어를 채택하여 시스템 내부의 탐색영역의 데이터를 저장하기 위한 부가적인 메모리를 제거하여 동작 속도 상의 이득뿐만 아니라, 메모리 공정을 필요로 하지 않는 저가의 공정을 사용 가능하게 함으로써 제작상의 비용을 절감할 수 있는 해결책을 제시하였다.

표 3. 움직임 추정기에 대한 VLSI 비교
Table. 3. Comparison of the proposed architecture with other existing architectures.

아키텍처	PE 수	처리량	탐색영역 범위	SRAM 사용여부
Natarajan [8]	16	1039	[-16,15]	○
Komarek [11]	256	496	[-8,7]	○
Hseih [12]	256	2209	[-8,7]	○
Yang [13]	16	4111	[-8,7]	○
Yeo [14]	1024	256	[-16,15]	○
Proposed	64	512	[-16,15]	×

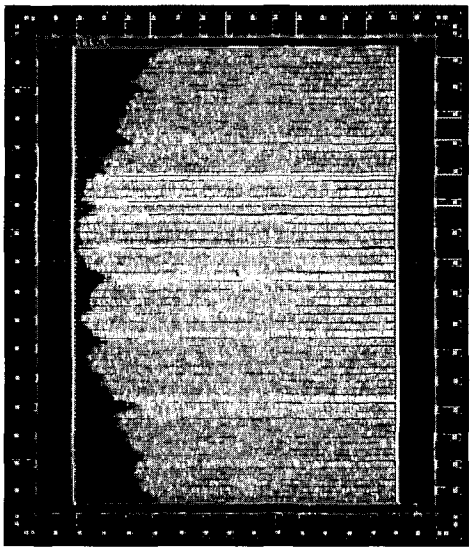


그림 7. 제안된 움직임 추정기에 대한 레이아웃
Fig. 7. Layout for the proposed motion estimator.

V. Conclusion

본 논문에서는 비트 플레인 정합 기준을 이용한 움직임 추정 알고리듬에 대한 움직임 추정기의 VLSI의 구조에 대하여 제시하였다. 제안된 움직임 추정기는 MAD(mean absulute difference)에 기반한 전역 탐색 알고리듬의 방대한 양의 계산량과 하드웨어로 구현했을 때 회로가 복잡해지는 문제점을 그레이 영상에서 추출한 하나의 이진 비트 플레인 영상에 적용함으로써 FSA와 유사한 움직임 추정 성능을 갖도록 하였다. 본 논문에서는 제안된 움직임 추정기의 정합기준은 단순한 이진 논리 연산만으로 처리되어, 정합기준을 계산하기 위한 하드웨어의 크기를 줄일 수 있는 방안을 제시하였다. 또한 제안된 VLSI 구조에서는 두 개의 프로세싱 코어를 채택하여 시스템 내부의 메모리를 제거하여 동작 속도 상의 이득뿐만 아니라, 메모리 공정을 필요로 하지 않는 저가의 공정을 사용 가능하게 하였으며, 실시간 움직임 추정 시스템과 같은 영상 압축 시스템의 스펙을 만족시킬 수 있는 방안에 대하여 설명하였다.

참 고 문 헌

[1] Video codec for audiovisual services at $p \times 64$ kbit/s, ITU-T Recommendation H.261, 1993.

[2] *Video coding for low bit rate communication*, ITU-T Recommendation H.263, 1996.

[3] ISO/IEC JTC1/SC29/WG11 13818-1, *Coding of moving pictures and associated audio*, Nov. 1994.

[4] J. R. Jain and A. K. Jain, "Displacement measurement and its application in interframe image coding," *IEEE Trans. Comm.*, vol. 29, no. 12, pp.1798-1808, Dec. 1981.

[5] Yunju Baek, Hwang-Seok Oh, and Heung-Kyu Lee, "Block-matching criterion for efficient VLSI implementation of motion estimation", *IEE Electronics Letters*, vol. 32, no. 13, pp. 1184-1185, Jun. 1996.

[6] Y.-K Ko, H.-G. Kim, H.-C. Oh, and S.-J. Ko, "Fast VLSI motion estimator based on bit plane matching," *IEE Electronics Letters*, vol. 36, no. 23, pp. 1923-1924, Nov. 2000.

[7] T. Koga, K. Iinuma, A. hirano, Y. Iijima, and T. Ishiguro, "Motion compression Interframe coding for video conferencing," *in Proc. Nat. Telecommnications Conf.*, New Orleans, LA, pp. G.5.3.1-G.5.3.5, Dec. 1981.

[8] B. Natarajan, V. Bhaskaran, and K. Konstantinides, "Low complexcity block-based motion estimation via one-bit transform," *IEEE Trans. Circuits Syst.*, Video Technol., vol. 7, no. 4, pp. 702-706, Aug. 1997.

[9] M. M. Mizuki, U. Y. Desai, I. Masaki, and A. Chandrakasan, "A binary block matching architecture with reduced power consumption and silicon area requirements," *IEEE ICASSP-96*, Atlanta, vol. 6, pp. 3248-3251, 1996.

[10] U.-V Koc and K. J. Ray Liu, "DCT-based motion estimation," *IEEE trans. Image Processing*, vol. 7, no. 7, pp. 948-4965, Jul. 1998.

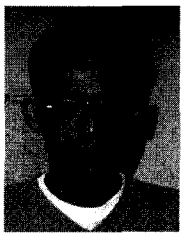
[11] T. Kormark and P. Pirsch, "Array architecture for block matching algorithm," *IEEE Circuits Syst.*, vol. 36, no. 10, pp. 1301-1308, Oct. 1989.

[12] C.-H. Heish and T.-P. Lin, "VLSI architecture for block matching motion estimation algorithm," *IEEE Trans. Circuits and Syst.*, Video Technol., vol. 2, no. 2, Jun. 1992.

[13] K.-M. Yang, M.-T. Sun, and L.-Wu, "A family of VLSI design for the motion compensation block matching algorithm," *IEEE trans. Circuits Syst.*, vol. 36, no. 10, pp. 1317-1325, Oct. 1989.

[14] H. Yeo and Y.-H Hu, "A novel modular systolic array architecture for full-search block matching motion estimation," *IEEE trans. Circuits Syst.*, Video Technol., vol. 5, no. 5, pp. 407-416, Oct. 1995.

저 자 소 개



高 永 基(正會員)

1995년 고려대학교 정보공학과 학사. 1997년 고려대학교 전자공학과 석사. 2001년 고려대학교 전자공학과 박사, 2001년 5월~현재, LG전자 Digital Media 연구소에서 "영상 캡쳐보드" 개발중

吳 亨 哲(正會員)

1982년 서울대학교 전자공학과 학사, 1984년 한국과학기술원 전기 및 전자공학과 석사. 1993년 Univ. of Maryland at College Park. 전기공학과 박사, 1984년~1987년 금성반도체 선임연구원, 1994년~현재 고려대학교 전자 및 정보공학부 부교수

高 聖 濟(正會員)

1980년 고려대학교 전자공학과 학사, 1986년 State Univ. of New York at Buffalo 전기 및 컴퓨터공학과 석사, 1988년 State Univ. of NewYork at Buffalo 전기 및 컴퓨터공학과 박사, 1981년 8월~1983년 12월 : 대한전선 중앙연구소 연구원, 1988년 8월~1992년 5월 : The Univ. of Michigan Dearborn, 전기 및 컴퓨터공학과 조교수, 1992년 3월~현재 : 고려대학교 전자공학과 교수, 1996년 11월 : IEEE APCCAS best paper award. 1997년 12월 : 대한전자공학회 해동논문상 수상, 1999년 11월 : 한국통신학회 LG 학술상 수상, IEEE Senior member, IEE Fellow.